Our Ref. No. 042390.P8097
Express Mail No.: EL466333747US

UNITED STATES PATENT APPLICATION

FOR

PLATFORM AND METHOD FOR ISSUING AND CERTIFYING A HARDWARE-PROTECTED ATTESTATION KEY

INVENTORS:

Carl M. Ellison
Roger A. Golliver
Howard C. Herbert
Derrick C. Lin
Francis X. McKeen
Gil Neiger
Ken Reneris
James A. Sutton
Shreekant S. Thakkar
Millind Mittal

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

BACKGROUND

1. Field

This invention relates to the field of data security. In particular, the invention relates to a platform and method for certifying a key within protected hardware.

2. Background

Advances in technology have opened up many opportunities for applications that go beyond the traditional ways of doing business. Electronic commerce (e-commerce) and business-to-business (B2B) transactions are now becoming popular, reaching the global markets at a fast rate. Unfortunately, while electronic platforms like computers provide users with convenient and efficient methods of doing business, communicating and transacting, they are also vulnerable for unscrupulous attacks. Examples of these attacks include virus, intrusion, security breach, and tampering, to name a few. Therefore, it is becoming more and more important to protect the integrity of data stored within or downloaded into a platform.

Various data security mechanisms may be used to protect the integrity of data exchanged between electronic platforms. One type of data security mechanism involves the development of cryptographic hardware having a private key stored in a secure manner. This hardware produces a digital signature by digitally signing data with the pre-stored private key in accordance with a selected digital signature function (e.g., Digital Signature Algorithm "DSA"). Accompanying the data during transmission, the digital signature protects the integrity of the data.

In order to recover the data, an authentication certificate normally accompanies the digital signature. The authentication certificate provides a public key corresponding to the private key for use in data recovery and for certifying (or attesting) to something. The meaning of a certificate depends on the contents of the certificate and the empowerment of the certificate signer (issuer). For example, the attestation may indicate that a given signature private key was installed in protected hardware, especially an isolated execution architecture designed below.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1A is a diagram illustrating an embodiment of the logical operating architecture for the ISOX™ architecture of the platform.

Figure 1B is an illustrative diagram showing the accessibility of various elements in the operating system and the processor according to one embodiment of the invention.

Figure 1C is a block diagram of an illustrative embodiment of a platform utilizing the present invention.

Figure 2 is a block diagram of an illustrative embodiment of the communications between the platform of Figure 1C and a remotely located platform.

Figure 3 is a block diagram of a first illustrative embodiment of the certification technique for the attestation key.

Figure 4 is a block diagram of a second illustrative embodiment of the certification technique for the attestation key.

Figure 5 is a block diagram of a third illustrative embodiment of the certification technique for the attestation key.

DETAILED DESCRIPTION

The present invention relates to a platform and method for certifying a key within protected hardware. Herein, certain details are set forth in order to provide a thorough understanding of the present invention. It is apparent to a person of ordinary skill in the art, however, that the present invention may be practiced through many embodiments other that those illustrated. Well-known circuits and hashing techniques are not set forth in detail in order to avoid unnecessarily obscuring the present invention.

In the following description, terminology is used to discuss certain features of the present invention. For example, a "platform" includes hardware equipment and/or software that perform different functions on stored information. Examples of a platform include, but are not limited or restricted to a computer (e.g., desktop, a laptop, a hand-held, a server, a workstation, etc.), desktop office equipment (e.g., printer, scanner, a facsimile machine, etc.), a wireless telephone handset, a television set-top box, and the like. A "software module" includes code that, when executed, performs a certain function. A "nub" is a series of code instructions, possibly a subset of code from a software module. A "link" is broadly defined as one or more information-carrying mediums (e.g., electrical wire, optical fiber, cable, bus, or wireless signaling technology).

In addition, the term "information" is defined as one or more bits of data, address, and/or control. A "segment" is one or more bytes of information. A "page" is a predetermined number of bytes, usually a power of two in length (e.g., 512, 1024, etc.). A "one-way hash function" is a function, mathematical or otherwise, that converts information from a variable-length to a fixed-length (referred to as a "hash value" or "digest"). The term "one-way" indicates that there does not readily exist an inverse function to recover any discernible portion of the original information from the fixed-length hash value. Examples of a hash function include MD5 provided by RSA Data

Security of Redwood City, California, or Secure Hash Algorithm (SHA-1) as specified a 1995 publication Secure Hash Standard FIPS 180-1 entitled "Federal Information Processing Standards Publication" (April 17, 1995).

I. Architecture Overview

A. Isolated Execution Platform

One principle for enhancing security of transmitted is through configuration of the platform with an isolated execution (ISOX™) architecture. The ISOX™ architecture includes logical and physical definitions of hardware and software components that interact directly or indirectly with an operating system of a platform. Herein, the operating system and a processor of the platform may have several levels of hierarchy, referred to as rings, which correspond to various operational modes. A "ring" is a logical division of hardware and software components that are designed to perform dedicated tasks within the operating system. The division is typically based on the degree or level of privilege, namely the ability to make changes to the platform. For example, a ring-0 is the innermost ring, being at the highest level of the hierarchy. Ring-0 encompasses the most critical, privileged components. Ring-3 is the outermost ring, being at the lowest level of the hierarchy. Ring-3 typically encompasses user level applications, which are normally given the lowest level of privilege. Ring-1 and ring-2 represent the intermediate rings with decreasing levels of privilege.

Figure 1A is a diagram illustrating an embodiment of a logical operating architecture 50 of the ISOX™ architecture. The logical operating architecture 50 is an abstraction of the components of an operating system and the processor. The logical operating architecture 50 includes ring-0 10, ring-1 20, ring-2 30, ring-3 40, and a processor nub loader 52. The logical operating architecture 50 has at least two modes of operation: normal execution mode and isolated execution mode. Each ring in the logical operating architecture 50 can operate in both modes. The processor nub loader 52 operates only in the isolated execution mode.

Ring-0 10 includes two portions: a normal execution Ring-0 11 and an isolated execution Ring-0 15. The normal execution Ring-0 11 includes software modules that are critical for the operating system. Typically, these software modules include a primary operating system 12 referred to as the "kernel"(e.g., the unprotected segments of the operating system), software drivers 13, and hardware drivers 14. The isolated execution Ring-0 15 includes an operating system (OS) nub 16 and a processor nub 18 as described below. The OS nub 16 and the processor nub 18 are instances of an OS executive (OSE) and processor executive (PE), respectively. The OSE and the PE are part of executive entities that operate in a secure environment associated with the isolated area 70 and the isolated execution mode. The processor nub loader 52 is a protected bootstrap loader code held within the chipset itself and is responsible for loading the processor nub 18 from the processor or chipset into a region of protected memory as further described below.

Similarly, ring-1 20, ring-2 30, and ring-3 40 include normal execution ring-1 21, ring-2 31, ring-3 41, and isolated execution ring-1 25, ring-2 35, and ring-3 45, respectively. In particular, normal execution ring-3 includes N applications 42_1-42_N and isolated execution ring-3 includes M applets 46_1-46_M (where "N" and "M" are positive whole numbers).

One concept of the isolated execution architecture is the creation of a region in system memory protected by the processor and /or chipset in the platform. This region of protected memory is referred to as an "isolated area". Access to the isolated area is permitted using special memory read and write cycles, which are referred to as "isolated read and write" cycles. The isolated read and write cycles are issued by the processor operating in the isolated execution mode.

The processor nub loader 52 verifies and loads a ring-0 nub software module (e.g., processor nub 18) into the isolated area. The processor nub 18 provides the basic hardware-related services to support isolated execution. For example, one task of the processor nub 18 is to verify and load the ring-0 OS nub 16 into the isolated area 70 as shown in Figure 1B.

The OS nub 16 provides links to services in the primary operating system 12, provides page management within the isolated area, and has the responsibility for loading some ring-0 software modules as well as ring-3 software modules 45 (e.g., applets 46_1-46_M) into protected pages allocated in the isolated area. The OS nub 16 may also support encrypting and hashing the isolated area pages before evicting the page(s) to the ordinary (unprotected) memory, and/or checking the page contents upon restoration of the page.

Figure 1B is an illustrative diagram showing the accessibility of various elements in the operating system 10 and the processor according to one embodiment of the invention. For clarity sake, only elements of ring-0 10 and ring-3 40 are shown. The various elements in the logical operating architecture 50 access an accessible physical memory 60 according to their ring hierarchy and the execution mode.

The accessible physical memory 60 includes an isolated area 70 and a non-isolated area 80. The isolated area 70 includes applet pages 72 and nub pages 74. The non-isolated area 80 includes application pages 82 and operating system (OS) pages 84. The isolated area 70 is accessible only to elements of the operating system and processor operating in isolated execution mode. The non-isolated area 80 is accessible to all elements of the ring-0 operating system and processor.

The normal execution ring-0 11 including the primary OS 12, the software drivers 13, and the hardware drivers 14, can access both the OS pages 84 and the application pages 82. The normal execution ring-3, including applications 42_1-42_N, can access only to the application pages 82. Both the normal execution ring-0 11 and ring-3 41, however, cannot access the isolated area 70.

The isolated execution ring-0 15, including the OS nub 16 and the processor nub 18, can access both the isolated area 70 (including the applet pages 72 and the nub pages 74) and the non-isolated area 80 (including the application pages 82 and the OS pages 84). The isolated execution ring-3 45, including applets 46_1-46_M, can access only the application pages 82 and the applet pages 72. The applets 46_1-46_M reside in the isolated area 70.

Referring to Figure 1C, a first block diagram of an illustrative embodiment of a platform utilizing the present invention is shown. The platform 100 comprises a processor 110, a system memory 140 and an input/output control hub (ICH) 150 in communication with each other. In this embodiment, however, the platform 100 further includes a memory control hub (MCH) 130 and a non-volatile memory (e.g., flash) 160 coupled to the ICH 150. The MCH 130 is further coupled to the processor 110 via a host bus 120. The ICH 150 may be integrated into a chipset together with or separate from the MCH 130.

It is contemplated that the platform 100 may be in communication with peripheral components such as a mass storage device 170, one or more input/output (I/O) devices 175, and a token 180 via a token bus 185 and/or a token reader 190. For clarity, the specific links for these peripheral components (e.g., Peripheral Component Interconnect "PCI", accelerated graphics port "AGP", Industry Standard Architecture "ISA", Universal Serial Bus "USB", etc.) are not shown.

The processor 110 represents a central processing unit of any type of architecture, such as complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In one embodiment, the processor 110 is compatible with the Intel Architecture (IA) processor, such as the IA-32 and the IA-64. The processor 110 includes an isolated execution circuit 115. The isolated execution circuit 115 provides a mechanism to allow the processor 110 to operate in an isolated execution mode. The isolated execution circuit 115 provides hardware and software support for the isolated execution mode. This support includes configuration for isolated execution, definition of the isolated area, definition (e.g., decoding and execution) of isolated instructions, generation of isolated access bus cycles, and generation of isolated mode interrupts.

The host bus 120 provides interface signals to allow the processor 110 to communicate with other processors or devices, e.g., the MCH 130. In addition to normal mode, the host bus 120 supports an isolated access bus mode with corresponding interface signals for isolated read and write cycles when the processor 110 is configured in the isolated execution mode. The isolated access bus mode is asserted on memory accesses initiated while the processor 110 is in the isolated execution mode if the physical address falls within the isolated area address range. The isolated access bus mode is also asserted on instruction pre-fetch and cache write-back cycles if the address is within the isolated area address range. The processor 110 responds to snoop cycles to a cached address within the isolated area address range if the isolated access bus cycle is asserted.

The MCH 130 provides control and configuration of memory and input/output devices such as the system memory 140 and the ICH 150. The MCH 130 provides interface circuits to recognize and service isolated access assertions on memory reference bus cycles, including isolated memory read and write cycles. In addition, the MCH 130 has memory range registers (e.g., base and length registers) to represent the isolated area in the system memory 140. Once configured, the MCH 130 aborts any access to the isolated area when the isolated access bus mode is not asserted.

The system memory 140 stores code and data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). The system memory 140 includes the accessible physical memory 60 (shown in Figure 1B). The accessible physical memory includes a loaded operating system (OS) 142, the isolated area 70 (shown in Figure 1B), and an isolated control and status space 148. The loaded OS 142 is the portion of the operating system that is loaded into the system memory 140. The loaded OS 142 is typically loaded from mass storage device 170 via some boot code in a boot storage such as a boot read only memory (ROM).

As shown in Figures 1B and 1C, the isolated area 70 is the memory area that is defined by the processor 110 when operating in the isolated execution mode. Access to the isolated area 70 is restricted and is enforced by the processor 110 and/or the MCH 130 or other chipset that integrates the isolated area functionalities.

Referring back to Figure 1C, the isolated control and status space 148 is an input/output (I/O)-like, independent address space defined by the processor 110 and/or the MCH 130. The isolated control and status space 148 contains (i) isolated execution control and status registers, and (ii) related initialization code invoked directly by the isolated instructions executed by the processor 110. The isolated control and status space 148 does not overlap any existing address space and is accessed using the isolated bus cycles. The system memory 140 may also include other programs or data that are not shown herein.

As shown in Figure 1C, the ICH 150 has a number of functionalities that are designed to support isolated execution in addition to the traditional I/O functions. In this embodiment, the ICH 150 comprises at least the processor nub loader 52 (shown in Figure 1A), a hardware-protected memory 152, a cryptographic key storage 154 and a token bus interface 158 to provide a signaling interface with the token bus 185. For clarity, only one ICH 150 is shown although platform 100 may be implemented with multiple ICHs. When there are multiple ICHs, a designated ICH is selected to control the isolated area configuration and status. This selection may be performed by an external strapping pin. As is known by one skilled in the art, other methods of selecting can be used.

The processor nub loader 52, as shown in Figures 1A and 1C, includes a processor nub loader code and its hash value (or digest). After being invoked by execution of an appropriated isolated instruction (e.g., ISO-INIT) by the processor 110, the processor nub loader 52 is transferred to the isolated area 70. Thereafter, the processor nub loader 52 copies the processor nub 18 from the non-volatile memory 160 into the isolated area 70, verifies and places the hash value of the processor nub 18 into an audit log of the hardware-protected memory 152 as described below. In one embodiment, the hardware-protected memory 152 is implemented as any memory array with single write, multiple read capability. This non-modifiable capability is controlled by logic or is part of the inherent nature of the memory itself. For example, as shown, the protected memory 152 may include a plurality of single write, multiple read registers.

As shown in Figures 1C, in one embodiment, the protected memory 152 includes an audit log 156. The “audit log” 156 is a listing of data that represents what information has been successfully loaded into the system memory 140 after power-on of the platform 100. For example, the representative data may be hash values of software modules loaded into the system memory 140. These software modules may include the processor nub 18, the OS nub 16, and/or any other critical software modules (e.g., ring-0 modules) loaded into the isolated area 70. Thus, the audit log 156 can act as a fingerprint that identifies information loaded into the platform (e.g., the ring-0 code controlling the isolated execution configuration and operation), and is used to attest or prove the state of the current isolated execution.

Of course, it is contemplated that the audit log 156 may be stored collectively in the protected memory 152 and unprotected memory (e.g., a memory array in the non-isolated area 80 of the system memory 140 of Figure 1B). For that case, a pointer to the beginning of a memory array in the unprotected memory is stored in the protected memory 152. In addition, the length of the audit log 156 and a total hash value of the contents of the audit log 156 are stored with the pointer as well.

The protected memory 152 further contains a predetermined address space reserved so that any read or write to the predetermined address space is passed to the token 180. For example, one portion of address space may include a register containing an address of an attestation key.

The cryptographic key storage 154 holds a symmetric key that is unique for the platform 100. In one embodiment, the cryptographic key storage 154 includes internal fuses that are programmed at manufacturing. Alternatively, the symmetric key contained in the cryptographic key storage 154 may create the aid of a random number generator.

Referring still to Figure 1C, the non-volatile memory 160 stores non-volatile information. Typically, the non-volatile memory 160 is implemented in flash memory. The non-volatile memory 160 includes the processor nub 18 and a binding key storage 164. The processor nub 18 provides the initial set-up and low-level management of the isolated area 70 of the system memory 140, including verification, loading, and logging of the OS nub 16, and the management of the symmetric key used to protect the operating system nub's secrets. The processor nub 18 may also provide application programming interface (API) abstractions to low-level security services provided by other hardware. The processor nub 18 may also be distributed by the original equipment manufacturer (OEM) or operating system vendor (OSV) via a boot disk.

The mass storage device 170 stores archive information such as code (e.g., processor nub 18), programs, files, data, applications (e.g., applications 42_1-42_N), applets (e.g., applets 46_1 to 46_M) and operating systems. The mass storage device 170 may also include platform readable medium such as a compact disk (CD) ROM 172 or a hard drive 176 as shown. Other types of "platform readable medium" include magnetic, optical or electrical medium such as a flash memory, an erasable ROM (EROM), a floppy diskette, an optical disk, and the like.

I/O devices 175 may include any I/O devices to perform I/O functions. Examples of I/O devices 175 include controller for input devices (e.g., keyboard, mouse, trackball, pointing device), media card (e.g., audio, video, graphics), communication card (e.g., network, modem, etc.) to establish a link with a remotely located platform, or any other peripheral controllers.

The token bus 185 provides an interface between the ICH 150 and one or more tokens 180 in the system. A "token" 180 is a device that performs dedicated I/O functions with security. The token 180 may be stationary (e.g., a motherboard token) or portable to be coupled via the token reader 190 as shown. The token bus interface 158 in the ICH 150 couples the token bus 180 to the ICH 150 and ensures that when commanded to prove the state of the isolated execution, the corresponding token 180 signs only valid isolated hash values.

B. Self-Generation and Communications

Referring to Figure 2, in one embodiment, an attestation key pair 200 is generated within the platform 100. A private attestation key (PRK) 210 of the attestation key pair 200 is stored in non-volatile memory protected by hardware. This non-volatile memory 215 may be part of the system memory 140 of Figure 1 or memory employed within the chipset such as the ICH 150 for example. It is contemplated, however, that a public attestation key 220 of the attestation key pair 200 is placed in non-volatile memory freely accessible by the software executed by the processor 110 or tokens 180 coupled to the token bus 185.

Upon receiving a challenge message 240 from a remotely located platform 250, the platform 100 generates a response message 260. Accompanied by a copy of the audit log 156, the response message 260 includes a random nonce 270 recovered from the challenge message 240 and a hash value 280 of the audit log. Since the hash value 280 may be modified during transit, software running in the isolated area 70 (see Fig. 1C) signs the random nonce 270 and hash value 280 with the private attestation key 210. Although not shown, the response message 260 may further include the audit log 156 and a certificate 290 including the public attestation key 220 and attesting that the private attestation key 210 was installed in hardware-protected memory. The audit log 156 and the certificate 290 would not need to be written into the signed portion of the message 260.

II. <u>Certification Techniques for the Self-Generated Attestation Key</u>

A. User Certification

Referring now to Figure 3, an illustrative flowchart of a first embodiment of the operations of the platform to certify a self-generated private attestation key loaded in protected memory is shown. During its first initial boot cycle that is normally trustable, the platform generates the attestation key pair, namely the public attestation key and its corresponding private attestation key (block 300). Herein, the attestation key pair may be generated by (i) a random or pseudo-random number generator possibly situated in the ICH, (ii) a token separate from the platform and coupled to the token bus and the like.

The user is presented with the public attestation key while the private attestation key is securely stored in memory, in one situation, by the private attestation key being stored in non-volatile memory within the platform (e.g., ICH). (blocks 310 and 320). Other situations include encrypting the private attestation key with a previously stored symmetric key. Examples for presentation of the public attestation key includes visually displaying the bits or characters forming the public attestation key or downloading the public attestation key to a platform readable medium for distribution to an entity having control of the remotely located platform for example. Thus, in certain relationships between the user and the entity (e.g., where the entity is configured to protect the user's data), the generation of the attestation key pair and providing of the public attestation key to the entity constitutes sufficient certification. That is, if the user is a reliable authority based on the perception of a certificate verifier (e.g., the entity), then the user can simply generate an attestation certificate. In this embodiment, the attestation certificate is information digitally signed by the private attestation key. Alternatively, the user can provide evidence to another entity to issue the attestation certificate.

B. Certification by an Agent for an Outside Entity

Referring now to Figure 4, an illustrative flowchart of a second embodiment of the operations of the platform to certify a self-generated attestation key is shown. For example, upon installation, upgrade or repair of the platform, an agent for an entity controlling a remotely located platform installs a platform readable medium (e.g., a token, CD-ROM, etc.) and boots the platform with code approved by the entity stored on the readable medium (blocks 400 and 410). The agent may now execute an applet running within the isolated area of the system memory to generate the attestation key pair (block 420).

In particular, a public attestation key of the attestation key pair is provided to an application supplied by the readable medium in order to produce an attestation certificate. The certificate generally includes the public attestation key encrypted with a private key of the entity represented by the agent (blocks 430 and 440). The public key and the certificate are placed in the protected memory for subsequent transmission to a remote platform when verifying the security of the platform (block 450). That is, when the user is not an acceptable authority on the state of the platform to some verifier, some agent for that verifier can inspect the platform and issue the attestation certificate.

C. Original Equipment Manufacturer

Referring to Figure 5, an illustrative flowchart of a third embodiment of the operations of the platform to certify a self-generated attestation key is shown. As set forth in block 500, the public attestation key is generated during assembly of the platform by the original equipment manufacturer (OEM). The OEM signs the certificate including the public attestation key to certify that its corresponding private key is securely stored in protected memory (e.g., either cryptographically secure or stored in hardware protected memory such as memory 152 of Figure 1C or in the isolated area 70) and that the protected memory has not been tampered with (blocks 510 and 520). This case is similar as described above, in that some agent other than the user is responsible for issuing the attestation certificate, but this case will have different cost and risk parameters.

In all three cases or any others like these, the choice of issuer of an attestation certificate depends on criteria of any entity that will verify the certificate. That choice is based on the verifier's own preexisting relationships with the various possible attesters: the platform owner, an agent who goes to the platform in question, the OEM who built the platform, etc. A single platform might have to satisfy multiple different verifiers and might therefore hold multiple attestation certificates, over one or more attestation keys.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.